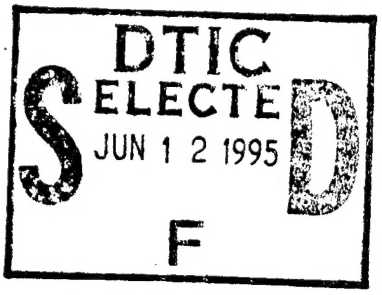
NAIC-ID(RS)T-0026-95

NATIONAL AIR INTELLIGENCE CENTER

C DECOMPILE CONTROLLING FLOW ANALYSIS--THE PROCESSING OF AN UNSTRUCTURED CODE PROGRAM

by

Lu Jiquan



HUMAN TRANSLATION

NAIC-ID(RS)T-0026-95 1 May 1995

MICROFICHE NR: 95C000277

C DECOMPILE CONTROLLING FLOW ANALYSIS--THE PROCESSING OF AN UNSTRUCTURED CODE PROGRAM

By: Lu Jiquan

English pages: 9

Source: Jisuanji Gongcheng, Vol. 18, Nr. 6, 1992, pp. 38-41

Country of origin: China
Translated by: SCITRAN
F33657-84-D-0165
Requester: NAIC/TATA/Keith D. Anthony

GRAPHICS DISCLAIMER

All figures, graphics, tables, equations, etc. merged into this translation were extracted from the best quality copy available.

C DECOMPILE CONTROLLING FLOW ANALYSIS--THE PROCESSING OF AN UNSTRUCTURED CODE PROGRAM

/38*

Lu Jiquan

ABSTRACT

This article analyses four basic unstructured models in program code as well as--during decompile controling flow analysis--analyzing the processing of these four types of unstructured models. This guarantees that high level program readability coming out of decompilation is good and maintains source program structure. This article also introduces translation to structured of main unstructured models as well as restoration of the C language characteristics of break and Continue and using GoTo statements to handle a number of unstructured situations.

KEY WORDS C Decompile Unstructured Models Controling Flow

Decompilation is a type of tool for translating low level code to high level code. As far as carrying out analysis of controling flow associated with low level code is concerned, it restores various types of control structures. For example, if-else structures, cyclical structures, Switch structures, and so on, are one key part in decompilation.

Low level codes--for example, compilation codes--are most certainly not all structured. The primary reason is that, in source programs, programmers utilize GoTo statements for their creation. Moreover, as far as unstructured factors contained in code to carry out targets produced after going through compilation are concerned, besides GoTo statements utilized by source programs, they are also given rise to by break and Continue statements in C language. If appropriate processing is not carried out on these unstructured factors in target code,

---

* Numbers in margins indicate foreign pagination. Commas in numbers indicate decimals.

then, during decompilation, it is very possible that the quality of translation results will be very bad. Because of this, as far as handling of unstructured problems in program flow graphs is concerned, it is very important in control flow analysis.

Processing of unstructured code includes two areas: the first, eliminating unstructured factors in program code; the second, maintaining as much as possible the structure of the source program. For example, break, Continue, and a number of necessary GoTo statements, etc.

1 PROGRAM CODE STRUCTURED TRANSLATION

Eliminating unstructured factors in program code then requires the carrying out of structured translations on them.

/39

According to the definitions of William MH., there are three types of basic structural flow graphs. They are, respectively, simple sequence, if-then-else, and while. These basic flow graphs are inserted into sets and build up into complex flow graphs. If one individual program graph is completely composed of these three types of basic structural flow charts, then the program graph is called structured. These three types of basic structured flow graph are as shown in Fig.1.

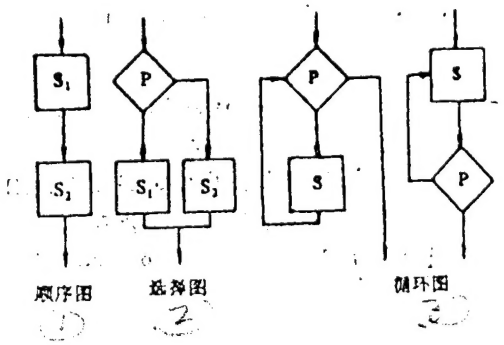


Fig.1

Key:(1) Simple Sequence (2) If-Then-Else (3) While

On the basis of the analytic induction of Oulsum G., there are four types of basic unstructured forms. They are ID, OD, IL, and OL as shown in Fig.2.

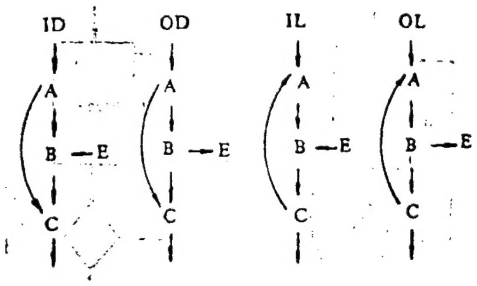


Fig.2

With regard to the realization of translations from unstructured to structured, it is only necessary to consider any three types from among the four types above. The reason is that any equivalent translation of unstructured models to structured models is in no case capable of occurring independently. The translation necessarily gives rise to a series of flow graph model translations. For example, processes for the elimination of the three types of ID, OD, and IL unstructured models in flow graphs on the basis of a fixed sequence will then cause OL unstructured models existing in source flow graphs to also be automatically eliminated. Therefore, we only need to consider equivalent translations associated with these three types of basic ID, OD, and IL unstructured models. These three types of translations are introduced below.

1.1 ID Structured Translations

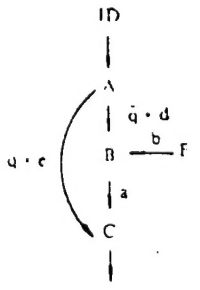


Fig.3

In this, q stands for a running direction (execute e) when decision point A satisfies condition q. When condition $\bar{q}$ is satisfied, after executing d, point B is arrived at. In the same way, b stands for operations executed during the process of taking nodal point E to B.

On the basis of Bambridge rules, Fig.3 translates to become Fig. 4.

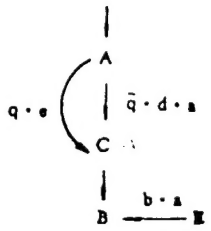


Fig.4

1.2 OD Structured Translation

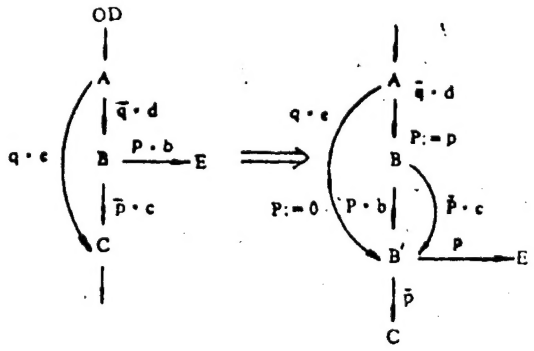


Fig.5

p.q are the conditions for decision points A and B. b, c, d, and e are operations. Introduce the new variable p. Add new operation P=0. P:=p and, taking the original point B shifts associated with p conditions, change them into shifts with regard to P. Also, add turning point B'shifts with regard to condition P.

1.3 IL Structured Translation

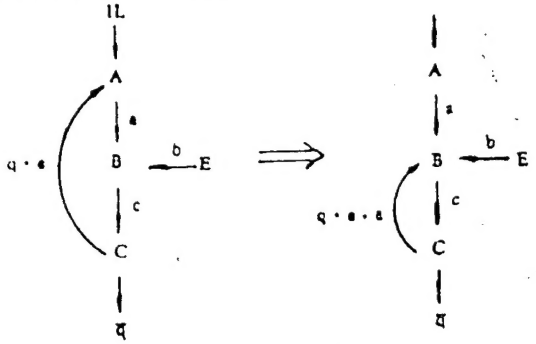


Fig.6

The meanings of symbols in this are the same as above. Structural translation seen in terms of equivalency is no problem. The reason is that each step of translation is mathematically equivalent. The entire (illegible) graph also maintains equivalency.

2 UNSTRUCTURED MODEL PROCESSING

With MicroVAXII/VMS operating system, we realized C decompilation systems. Processing principles for unstructured codes in program code are: eliminate the main unstructured factors, and, as much as possible, maintain original program code structures, strengthening easy readability. Below, unstructured code processing based on program graphs is introduced.

2.1 Unstructured Code Processing Associated with ID

Due to the fact that our C decompilation system carries out control flow analysis based on program graph methods, reference was made to the whole article "Decompilation Program Graph Design and Control Flow Analysis". Therefore, it is possible not to consider it for this unstructured type. The reason is that, as far as this type of unstructured model is concerned, in program graphs, automatic elimination is possible.

2.2 Unstructured Processing Associated with OD

With regard to unstructured jump outs from decisions, in programs in general, there may still be some. Programmers are able to use GoTo statements stemming from certain types of special causes. As a result, during processing, we also did not do structured translations, but used GoTo statements for processing. Fig.7 then shows program flow after processing.

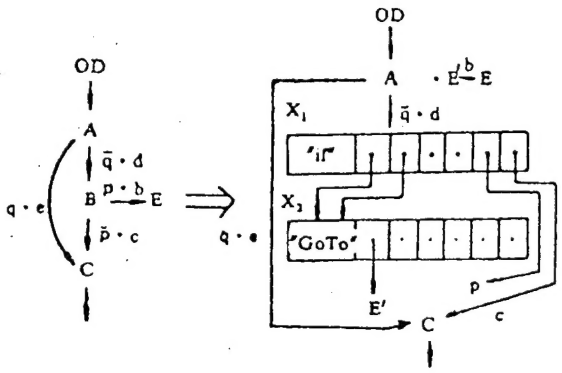


Fig.7

In the case of the newly added junction points x1, x2 in flow diagrams, if statements and GoTo statements are used together to handle unstructured, and operation b taken to point B is separated out from the flow graph. As far as flow graphs from E' to E are concerned, unstructured code processing is handled in the same way.

2.3 Unstructured Processing Associated with L

Treatment of unstructured code which jumps out of loops is divided into three situations for processing: (1) using break statements for processing, (2) using Continue statements for processing, and (3) using GoTo statements for processing.

With break statements, it is possible to eliminate numerous loop outlets. The same as 2.2, use is also made of GoTo

statement processing. Respective introductions are made below. Loops use dowhile forms as examples. while forms are similar--as is shown in Fig.8.

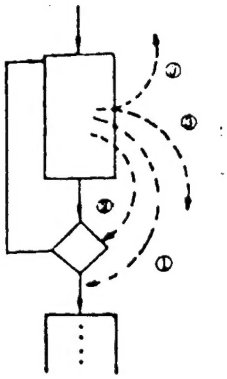


Fig.8

Dotted lines show three situations associated with jump outs from loops. (1) Restoring break statement and eliminating unstructured elements. (2) Restoring Continue statement and eliminating unstructured elements. (3) Using GoTo statements to process unstructured code.

(1) Use break statements to eliminate unstructured code (Fig.9)

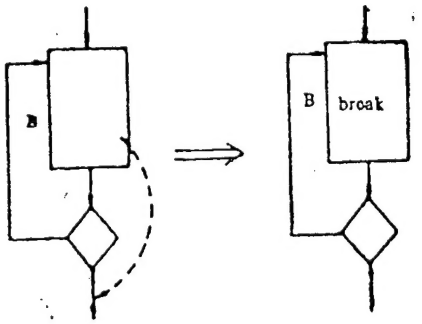


Fig.9

(2) Use Continue statements to eliminate overlapping loops (Fig.10)

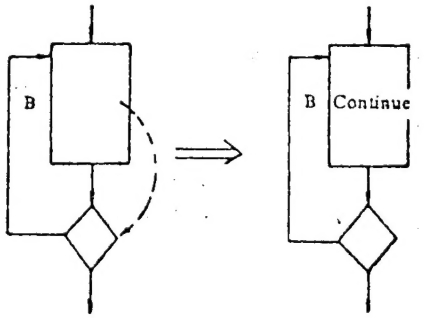


Fig.10

/41

(3) Use GoTo statements to process unstructured code (Fig.11)

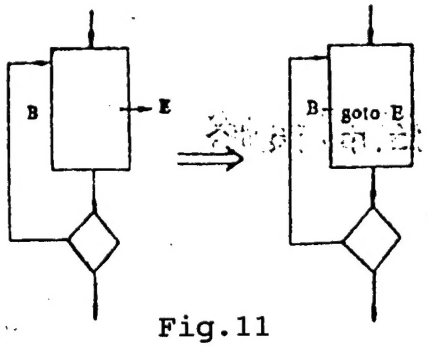


Fig.11

2.4 Unstructured Processing Associated with IL

Structured transformations must be carried out with regard to unstructured codes jumping into loops. Because using GoTo statements to jump into loops is very rarely seen, programmers generally do not use them. Moreover, some high level languages basically do not permit this type of situation to appear.

Assume Q is a logic variable. Below is a display of IL structured translation as shown in Fig.12.

Add the two expressions Q:=1, Q:=0.

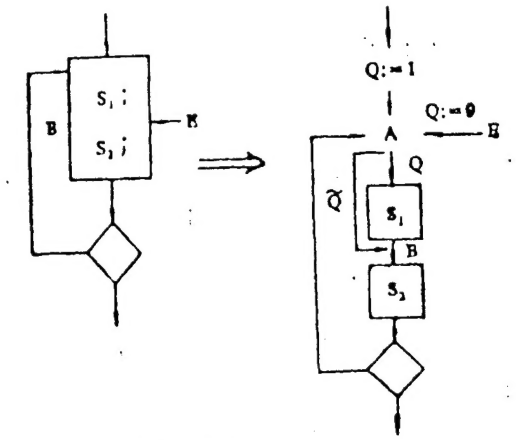


Fig.12

Point A adds conditions to a decision.

REFERENCES

1 Oulsnam G. Unravelling Unstructured Programs. The Computer Journal 1982 25 (3): 379

2 Oulsnam G. The Algorithmic Transformation of Schemas to Structured Form. The Computer Journal 1987. 30 (1): 43

(Editor Zhang Tingjun)

DISTRIBUTION LIST

DISTRIBUTION DIRECT TO RECIPIENT

ORGANIZATION	MICROFICHE
B085 DIA/RTS-2FI	1
C509 BALLOC509 BALLISTIC RES LAB	1
C510 R&T LABS/AVEADCOM	1
C513 ARRADCOM	1
C535 AVRADCOM/TSARCOM	1
C539 TRASANA	1
Q592 FSTC	4
Q619 MSIC REDSTONE	1
Q008 NTIC	1
Q043 AFMIC-IS	1
E051 HQ USAF/INET	1
E404 AEDC/DOF	1
E408 AFWL	1
E410 AFDTC/IN	1
E429 SD/IND	1
P005 DOE/ISA/DDI	1
P050 CIA/OCR/ADD/SD	2
1051 AFIT/LDE	1
P090 NSA/CDB	1
2206 FSL	1

Microfiche Nbr: FTD95C000277
NAIC-ID(RS)T-0026-95